und.1 Máquinas de Turing Universais

tur:und:uni:sec

Na **??** discutimos como toda máquina de Turing pode ser descrita por uma sequência finita de inteiros. Essa sequência codifica os estados, o alfabeto, o estado inicial e as instruções da máquina de Turing. Também observamos que o conjunto de todas essas descrições é enumerável. Como o conjunto de tais descrições é infinito enumerável, isso significa que há uma função sobrejetiva de $\mathbb{N}$ nessas descrições. Uma tal função sobrejetiva pode ser obtida, por exemplo, usando o método zigue-zague de Cantor. Ele nos dá uma maneira de enumerar todas as (descrições de) máquinas de Turing. Se fixarmos uma tal enumeração, passa a fazer sentido falar da primeira, da segunda, ..., e-ésima máquina de Turing. Esses números são chamados de *índices*.

Definição und.1. Se M é a e-ésima máquina de Turing (em nossa enumeração fixada), dizemos que e é um *índice* de M. Escrevemos M_e para a e-ésima máquina de Turing.

Uma máquina pode ter mais de um índice; por exemplo, duas descrições de M podem diferir na ordem em que listamos suas instruções, e essas diferentes descrições terão índices diferentes.

É importante notar que é possível dar a enumeração das descrições de máquinas de Turing de tal modo que possamos computar efetivamente a descrição de M a partir de seu índice, e computar efetivamente um índice de uma máquina M a partir de sua descrição. Pela tese de Church-Turing, é então possível encontrar uma máquina de Turing que recupera a descrição da máquina de Turing com índice e e escreve a descrição correspondente em sua fita como saída. A descrição seria uma sequência de blocos de 1 (representando os inteiros positivos na sequência que descreve M_e).

Diante disso, torna-se natural perguntar: que funções de índices de máquinas de Turing são elas próprias computáveis por máquinas de Turing? Que propriedades de índices de máquinas de Turing podem ser decididas por máquinas de Turing? Um exemplo: a função que leva um índice e ao número de estados que a máquina de Turing com índice e possui é computável por uma máquina de Turing. Eis o que tal máquina de Turing faria: iniciada em uma fita contendo um único bloco de e 1, ela primeiro decodificaria e em sua descrição. A descrição passa a ser representada por uma sequência de blocos de 1 na fita. Como o primeiro membro nessa sequência é o número de estados, tudo o que resta a fazer agora é apagar tudo, exceto o primeiro bloco de 1, e então parar.

Um resultado notável é o seguinte:

tur:und:uni:thm:universal-tm

Teorema und.2. *Há uma* máquina de Turing universal U *que, quando iniciada na entrada* $\langle e, n\rangle$*,*

1. *para se, e somente se,* M_e *para na entrada* n*, e*
2. *se* M_e *para com saída* m*, então* U *também para.*

Assim, U computa a função $f\colon \mathbb{N}\times\mathbb{N} \nrightarrow \mathbb{N}$ dada por $f(e,n) = m$ se M_e iniciada na entrada n para com saída m, e indefinida caso contrário.

Prova. Produzir de fato U é basicamente impossível, já que se trata de uma máquina extremamente complicada. Mas podemos descrever em linhas gerais como ela funciona, e então invocar a tese de Church-Turing. Quando ela começa, a fita de U contém um bloco de e 1 seguido de um bloco de n 1. Ela primeiro "decodifica" o índice e à direita da entrada n. Isso produz uma lista de números (isto é, blocos de 1 separados por 0) que descreve as instruções da máquina M_e. Em seguida, U escreve o número do estado inicial de M_e e o número 1 na fita, à direita da descrição de M_e. (Novamente, esses são representados em unário, como blocos de 1.) Depois, ela copia a entrada (bloco de n 1) para a direita—mas substitui cada 1 por um bloco de três 1 (lembre-se, o número do símbolo 1 é 3, sendo 1 o número de ▷ e 2 o número de 0). Na extremidade esquerda dessa sequência de blocos (separados por símbolos 0 na fita de U), ela escreve um único 1, o código de ▷.

U agora tem em sua fita: o índice e, o número n, o número de código do estado inicial (o "estado atual"), o número da posição inicial do cabeçote 1 (a "posição atual do cabeçote") e o conteúdo inicial da "fita" (uma sequência de blocos de 1 representando os números de código dos símbolos de M_e—os "símbolos"—separados por 0).

Ela agora simula o que M_e faria se iniciada na entrada n, fazendo o seguinte:

1. Encontrar o número k da "posição atual do cabeçote" (no início, esse é 1),
2. Mover-se para o k-ésimo bloco na "fita" para ver qual é o "símbolo" ali,
3. Encontrar a instrução que corresponde ao "estado" e ao "símbolo" atuais, tur:und:uni: find-inst
4. Voltar para o k-ésimo bloco na "fita" e substituir o "símbolo" ali pelo número de código do símbolo que M_e escreveria,
5. Mover o cabeçote para onde ela registra o "estado" atual e substituir o número ali pelo número do novo estado,
6. Mover-se para o lugar onde ela registra a "posição na fita" e apagar um 1 ou adicionar um 1 (se a instrução mandar mover para a esquerda ou para a direita, respectivamente).
7. Repetir.[1]

Se M_e iniciada na entrada n nunca para, então U também nunca para, de modo que sua saída é indefinida.

[1] Estamos passando por cima de algumas dificuldades sutis aqui. Por exemplo, U pode precisar de algum espaço extra quando incrementa o contador onde mantém o registro da "posição atual do cabeçote"—nesse caso, terá de mover toda a "fita" para a direita.

Se no passo (3) acontecer de a descrição de M_e não conter nenhuma instrução para o par "estado"/"símbolo" atual, então M_e pararia. Se isso acontecer, U apaga a parte de sua fita à esquerda da "fita". Para cada bloco de três 1 (representando um 1 na fita de M_e), ela escreve um 1 na extremidade esquerda de sua própria fita, e apaga sucessivamente a "fita". Quando isso está concluído, a fita de U contém um único bloco de 1 de comprimento m.

Se U encontrar algo diferente de um bloco de três 1 na "fita", ela para imediatamente. Como a fita de U nesse caso não contém um único bloco de 1, sua saída não é um número natural, isto é, $f(e, n)$ é indefinida nesse caso. □

Créditos das Imagens

Referências Bibliográficas